

CSE 493s/599s

Lecture 7. Alignment



Sewoong Oh



Lecture notes

- These lecture notes are based on other courses in LLMs, including
 - CSE493S/599S at UW by Ludwig Schmidt: <https://mlfoundations.github.io/advancedml-sp23/>
 - EE-628 at EPFL by Volkan Cevher: <https://www.epfl.ch/labs/lions/teaching/ee-628-training-large-language-models/ee-628-slides-2025/>
 - ECE381V Generative Models at UT Austin by Sujay Sanghavi
 - and various papers and blogs cited at the end of the slide deck

Outline

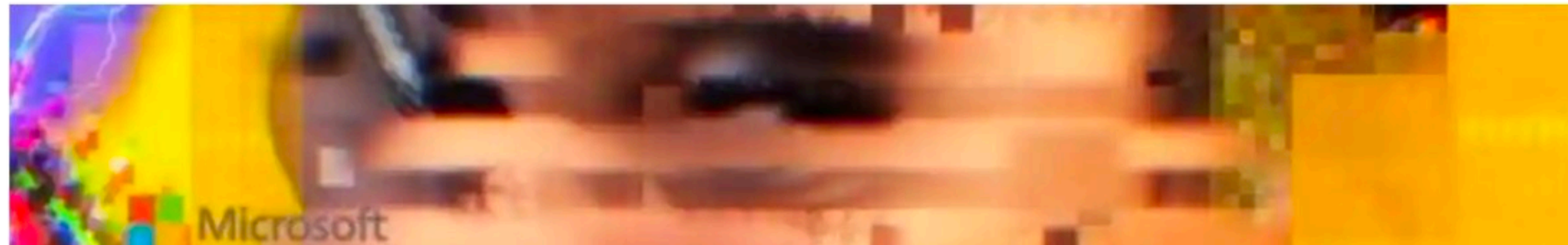
- **Tokenizers**
- **Language models**
- **Architecture**
 - **Transformers**
 - **Mixture-of-experts**
- **Inference**
 - **Speculative decoding**
 - **In-context learning**
 - **Chain-of-thought prompting**
 - **Test-time compute**
- **Post-training**
 - **Parameter Efficient fine-tuning**
 - **Alignment**

Alignment

- AI model responses can be misaligned with what we want them to do, which can sometimes cause real harm.

Microsoft 'deeply sorry' for racist and sexist tweets by AI chatbot

Company finally apologises after 'Tay' quickly learned to produce offensive posts, forcing the tech giant to shut it down after just 16 hours



The New York Times

Artificial Intelligence > A.I. Faces Quiz How the A.I. Race Began Key Figures in the Field One Year of ChatGPT

User: Create animated toys

A.I.: GENERATED BY A.I. [Image of a copyrighted Disney-style animated scene]

We Asked A.I. to Create the Joker. It Generated a Copyrighted Image.

By Stuart A. Thompson Jan. 25, 2024

Chatbots' inaccurate, misleading responses about U.S. elections threaten to keep voters from polls

By Garance Burke | AP
February 27, 2024 at 5:07 p.m. EST

The Washington Post
Democracy Dies in Darkness

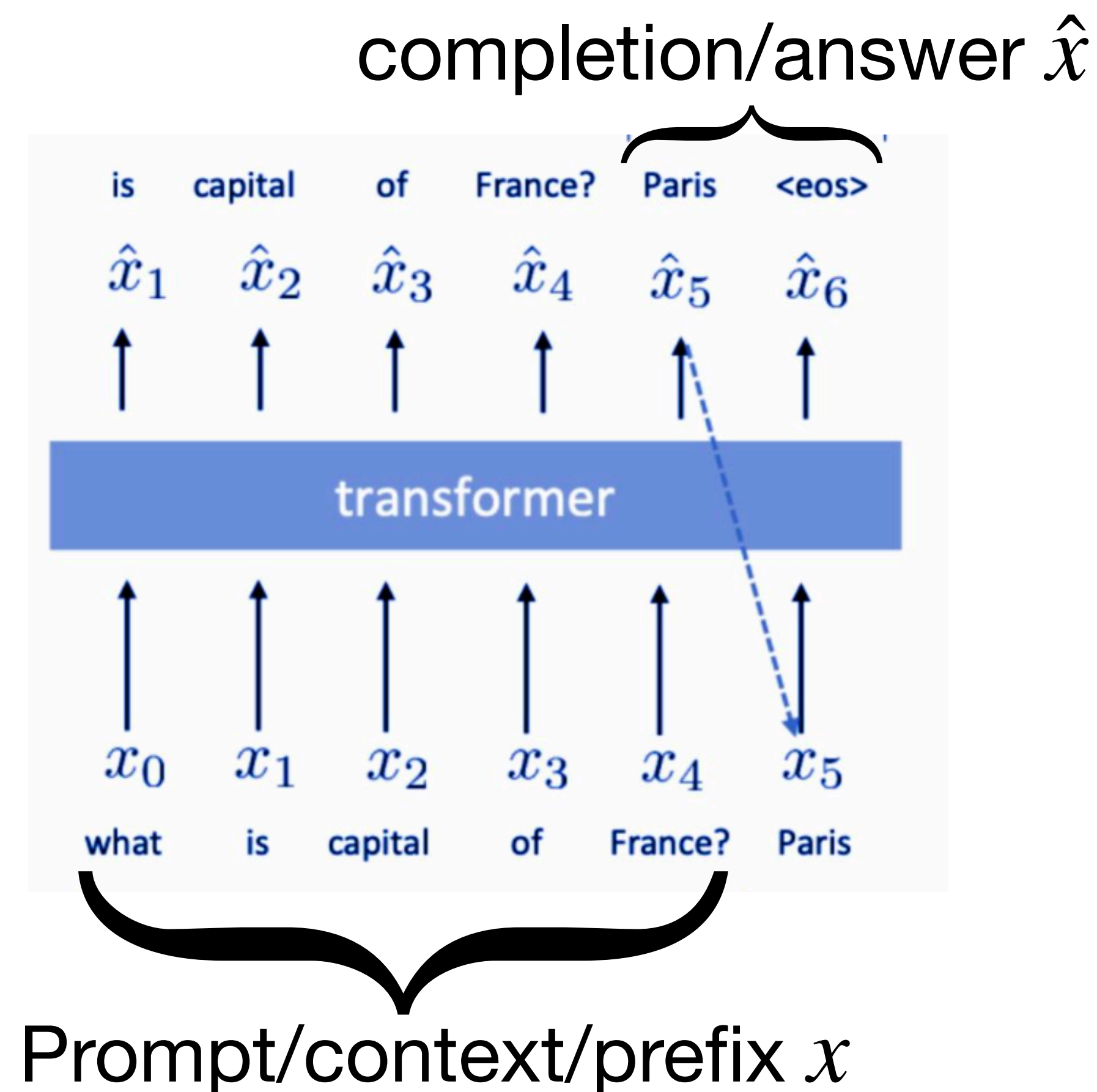
- Scaling pertaining does not address the challenges in **alignment** with human values and intent. We need post-training based on **RLHF (Reinforcement Learning with Human Feedback)**.
- We do not cover reinforcement learning in this class in any depths, but we will learn as much as we need along the way.

- Given a prompt/context/prefix x and its completion \hat{x} , a **reward model** assigns a scalar value on the quality of the completion:

$$r(x, \hat{x}) \in \mathbb{R}$$

- In RL, the language model is called a **policy** that assigns a probability to a completion:

$$\pi(\hat{x} | x)$$



- RLHF to train an LM to write summaries better.

1 Collect human feedback

A Reddit post is sampled from the Reddit TL;DR dataset.



Various policies are used to sample a set of summaries.



Two summaries are selected for evaluation.



A human judges which is a better summary of the post.



"j is better than k"

2 Train reward model

One post with two summaries judged by a human are fed to the reward model.



The reward model calculates a reward r for each summary.



r_j

r_k

The loss is calculated based on the rewards and human label, and is used to update the reward model.

$$\text{loss} = \log(\sigma(r_j - r_k))$$

"j is better than k"

3 Train policy with PPO

A new post is sampled from the dataset.



The policy π generates a summary for the post.



The reward model calculates a reward for the summary.



The reward is used to update the policy via PPO.



r

Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

- Comparing human evaluation of various summarization techniques:

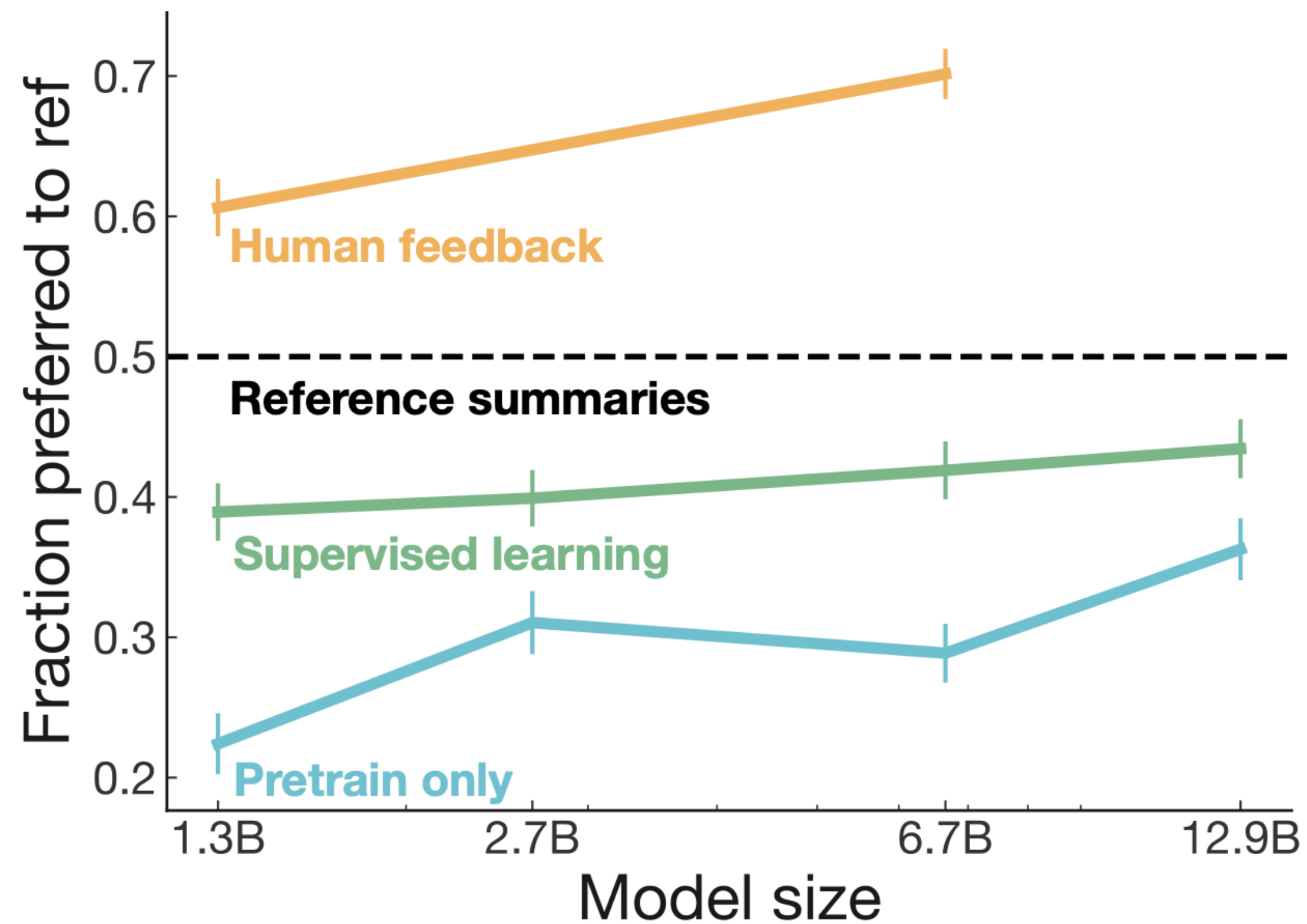
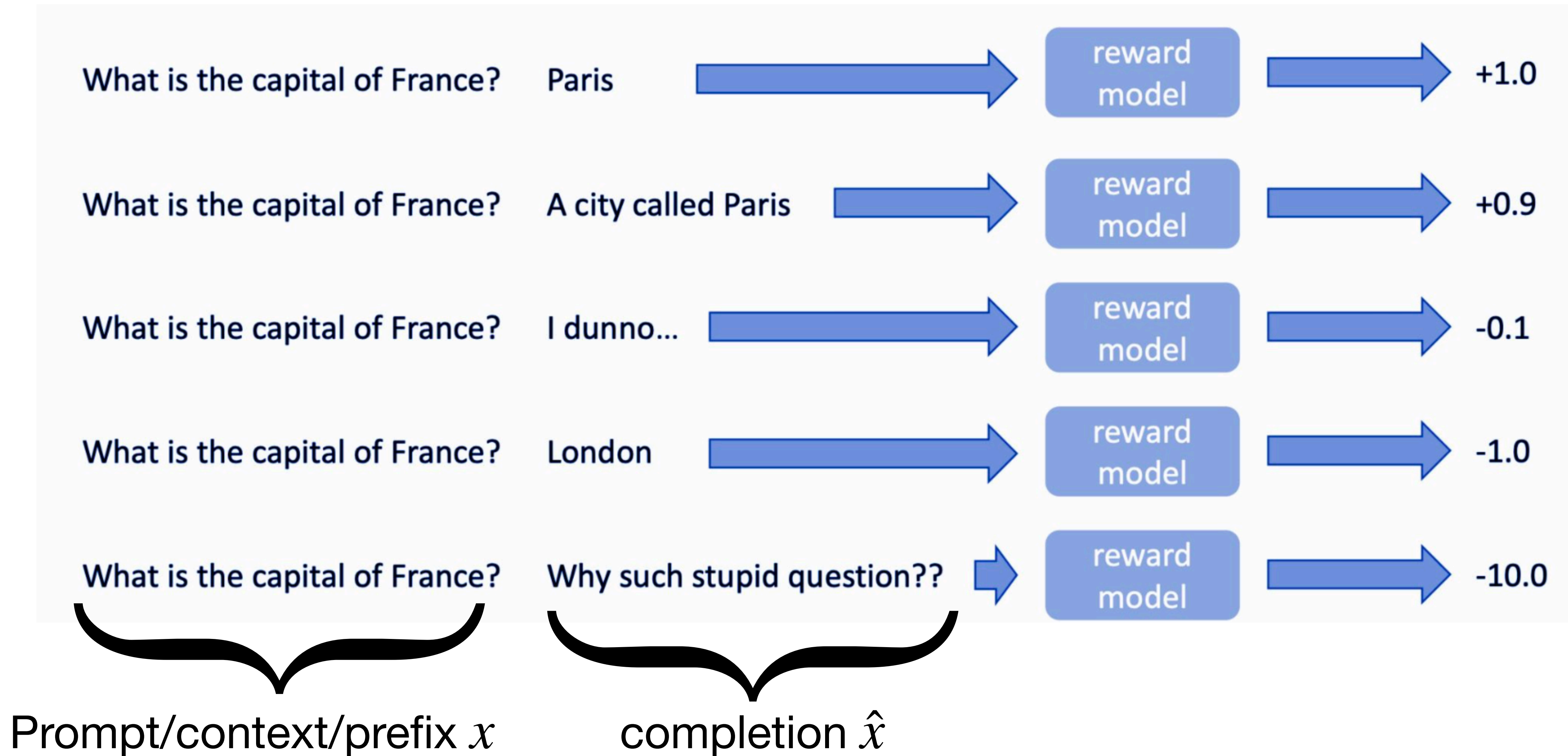


Figure 1: Fraction of the time humans prefer our models’ summaries over the human-generated reference summaries on the TL;DR dataset.⁴Since quality judgments involve an arbitrary decision about how to trade off summary length vs. coverage within the 24-48 token limit, we also provide length-controlled graphs in Appendix F; length differences explain about a third of the gap between feedback and supervised learning at 6.7B.

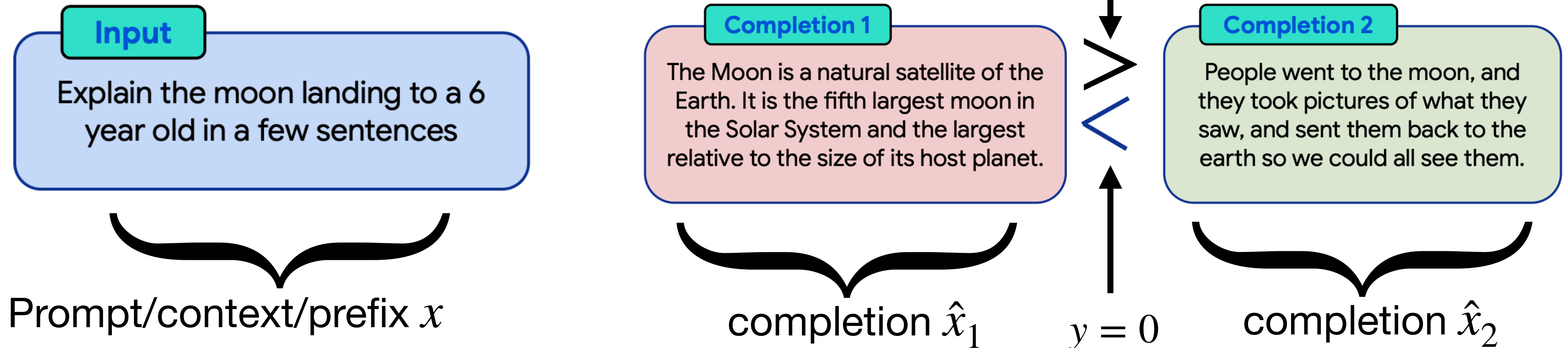
- **RLHF:** Strategy for further training the LM to make it aligned.
 1. Pretrain a language model.
 2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value
 3. Fine-tune a pre-trained LM with standard Reinforcement Learning (RL) using the reward model.
- This is one example of RLHF and there are several difference variations, including Direct Preference Optimization (DPO) and Group Relative Policy Optimization (GRPO).

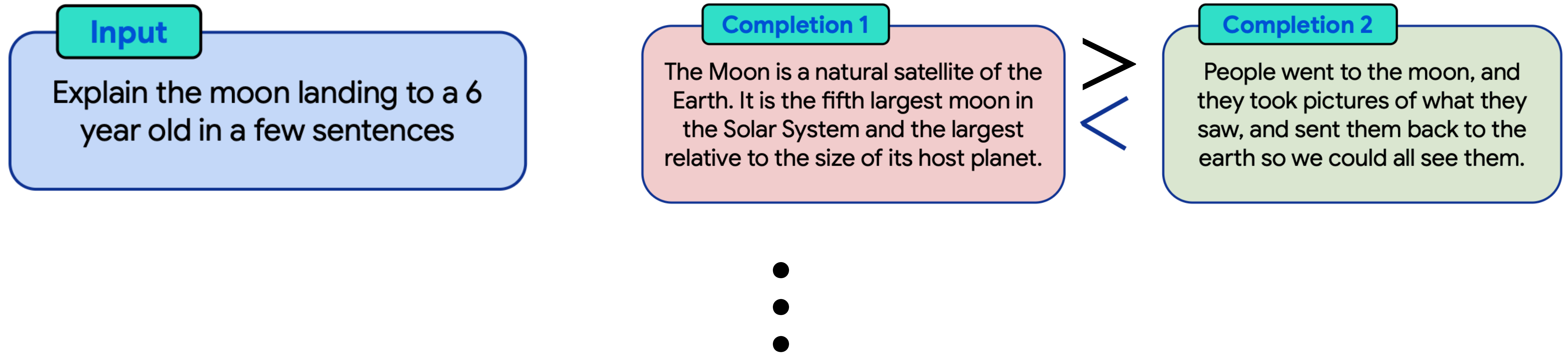
- **RLHF:** Strategy for further training the LM to make it aligned.
 1. Pretrain a language model.
 2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value
 3. Fine-tune a pre-trained LM with standard Reinforcement Learning (RL) using the reward model.
- This is one example of RLHF and there are several difference variations, including Direct Preference Optimization (DPO) and Group Relative Policy Optimization (GRPO).

- consider a neural network function $r(x, \hat{x})$, that we want to train on **human labelled preference**, such that it provides useful “reward”
 - but getting numerical reward from human is not easy

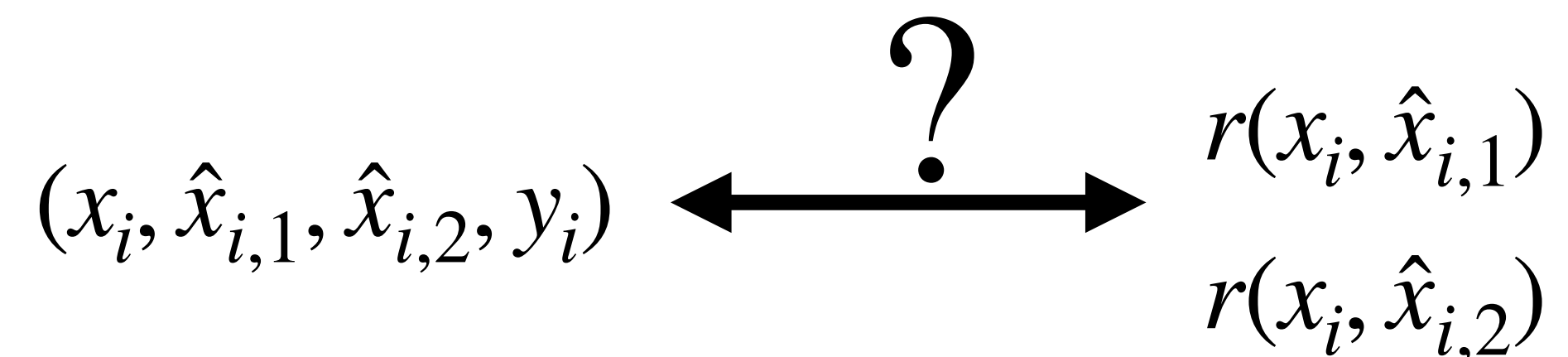


- However, it is challenging to collect reliable reward, so instead we collect data consisting of **pairwise comparisons** on which completion is preferred. This does not fit questions where exact solutions exist, like math and coding, where we use **RLVR (Reinforcement Learning with Verifiable Reward)** instead.
- Each sample $(x, \hat{x}_1, \hat{x}_2, y)$ consists of
 - prompt x ,
 - two different completions \hat{x}_1 and \hat{x}_2 , and
 - preference label $y = 1$ if \hat{x}_1 is preferred over \hat{x}_2 , which we write as $\hat{x}_1 \succ \hat{x}_2$, and $y = 0$, otherwise.





- Given a dataset of $\{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i)\}$, we need a **mathematical model** to learn the associated reward values $r(x_i, \hat{x}_{i,1})$ and $r(x_i, \hat{x}_{i,2})$: how do we associate the observed preference y_i to the hidden rewards?



Choice modeling

- Long before internet or computers, people were interested in solving the following optimization:

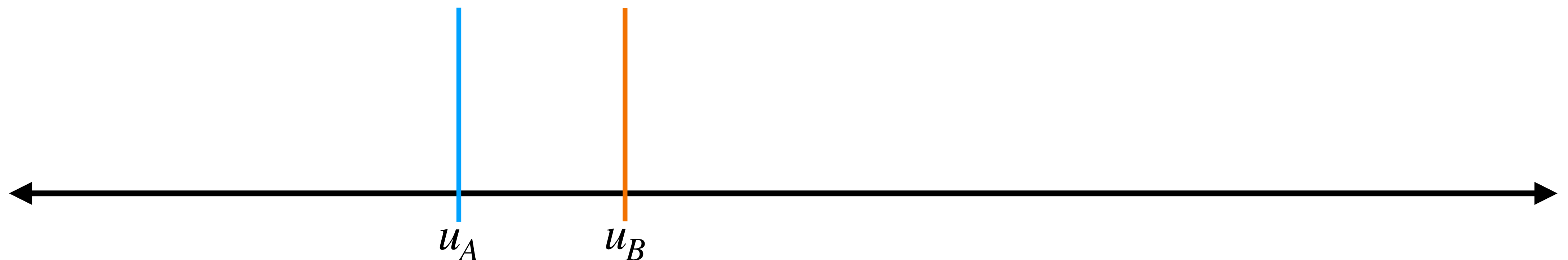
$$\mathit{max}_{\text{a set of items } S} \sum_{i \in S} \text{Profit}_i \times P(i | S)$$

- this requires inferring the choice probability from historical purchase data

$$P(i | S) = \mathbb{P} \left(\text{the next customer chooses an item } i \mid \text{a set of items } S \text{ is displayed} \right)$$

- People turned to mathematical modeling for help.

- So, the mathematical foundations of **choice models**, in particular, **Random Utility Models (RUMs)**, is introduced.
- Under RUM, each option has corresponding **hidden, inherent value/utility**, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility.
 - **Example:** Given two items to choose from **{A,B}**, **RUMs** assume that
 - the two items have inherent hidden value called **utility**: u_A, u_B



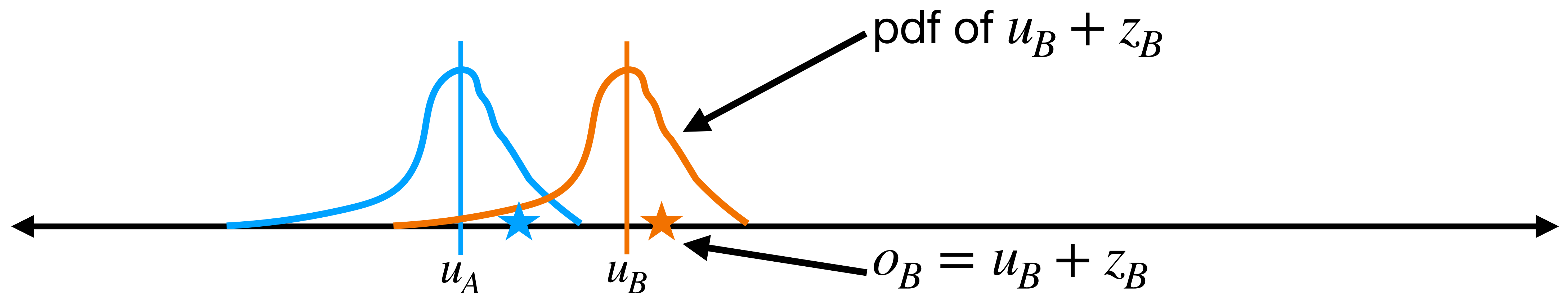
- So, the mathematical foundations of **choice models**, in particular, **Random Utility Models (RUMs)**, is introduced.
- Under RUM, each option has corresponding **hidden, inherent value/utility**, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility.

- **Example:** Given two items to choose from $\{A, B\}$, **RUMs** assume that

- the two items have inherent hidden value called **utility:** u_A, u_B

- the customer observes noisy version of the utilities, **observed utility:**

$$o_A = u_A + z_A, \quad o_B = u_B + z_B$$



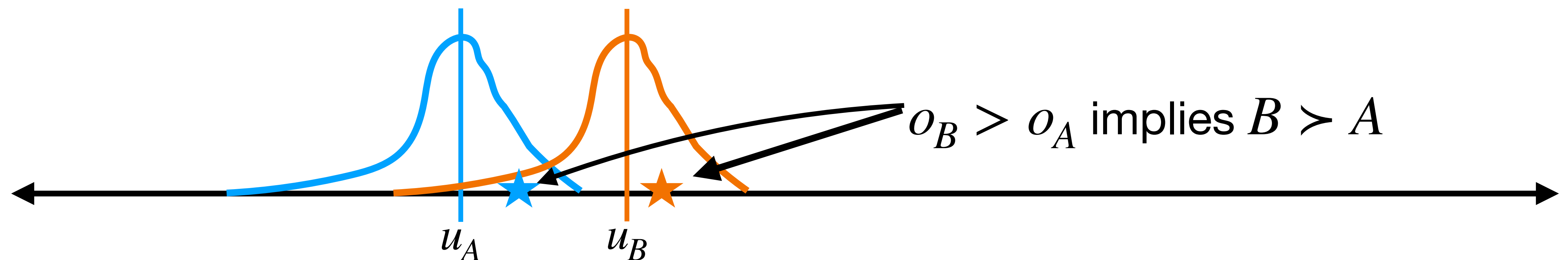
- So, the mathematical foundations of **choice models**, in particular, **Random Utility Models (RUMs)**, is introduced.
- Under RUM, each option has corresponding **hidden, inherent value/utility**, and when we make a choice, we observe a randomly perturbed utility and choose the one that has maximum observed utility.

- **Example:** Given two items to choose from $\{A, B\}$, **RUMs** assume that

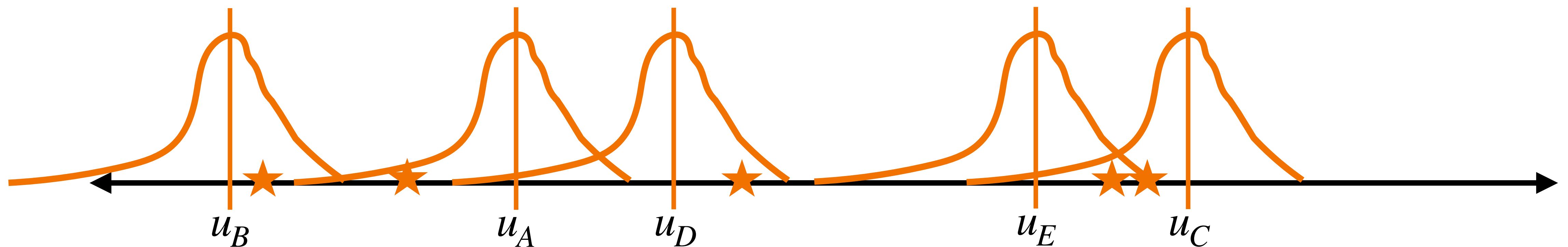
- the two items have inherent hidden value called **utility**: u_A, u_B
- the customer observes noisy version of the utilities, **observed utility**:

$$o_A = u_A + z_A, \quad o_B = u_B + z_B$$

- the customer's **choice** is determined by which option has higher observed utility.



- So, the mathematical foundations of **choice models**, in particular, **Random Utility Models (RUMs)**, is introduced.
- This mathematical framework naturally extends to any number of items, allowing one to
 - learn the hidden utility/reward u_A, \dots from historical purchase data
 - make predictions on future customers
- $P(i | S) = \mathbb{P}(\text{the next customer chooses an item } i \mid \text{a set of items } S \text{ is displayed})$
 $= \text{SoftMax}([u_A, u_B, u_C, \dots])_i$
 $= \frac{e^{u_i}}{e^{u_A} + e^{u_B} + \dots}$



- Formally, **Random Utility Model (RUM)** for a given prompt x , and two completions \hat{x}_1 and \hat{x}_2 is defined as
 - hidden true **rewards (=utility)** of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
 - **observed rewards**: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$
 - **preference (=choice)**: $\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$

$$= \mathbb{P}\left(\underbrace{z_1 - z_2}_{\text{some R.V.}} > \underbrace{r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)}_{\text{utility difference}} \right)$$
- note that the outcome only depends on the difference of the true rewards: $r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)$
- and the distribution of the noise z 's defined the probability distribution.

- Formally, **Random Utility Model (RUM)** for a given prompt x , and two completions \hat{x}_1 and \hat{x}_2 is defined as
 - hidden true **rewards (=utility)** of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
 - observed rewards**: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$
 - preference (=choice)**: $\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$

$$= \mathbb{P}\left(\underbrace{z_1 - z_2}_{\text{some R.V.}} > \underbrace{r^*(x, \hat{x}_2) - r^*(x, \hat{x}_1)}_{\text{utility difference}} \right)$$

$$= \frac{1}{1 + \exp\{- (r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2))\}}$$

$$= \sigma(r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2))$$

- Different choices of the noise gives different models. When the noise z_i 's follow independent **Gumbel distribution**, the resulting distribution of the preference simplifies to a **sigmoid function**, which is called **Bradley-Terry model**.

- **Bradley-Terry (BT) model** or Bradley-Terry-Luce (BTL) model

- hidden true rewards of the two completions: $r^*(x, \hat{x}_1)$ and $r^*(x, \hat{x}_2)$
- observed rewards: $r^*(x, \hat{x}_1) + z_1$ and $r^*(x, \hat{x}_2) + z_2$

- **preference:** when $y = 1$,

$$\mathbb{P}(\hat{x}_1 \succ \hat{x}_2) = \mathbb{P}(r^*(x, \hat{x}_1) + z_1 > r^*(x, \hat{x}_2) + z_2)$$

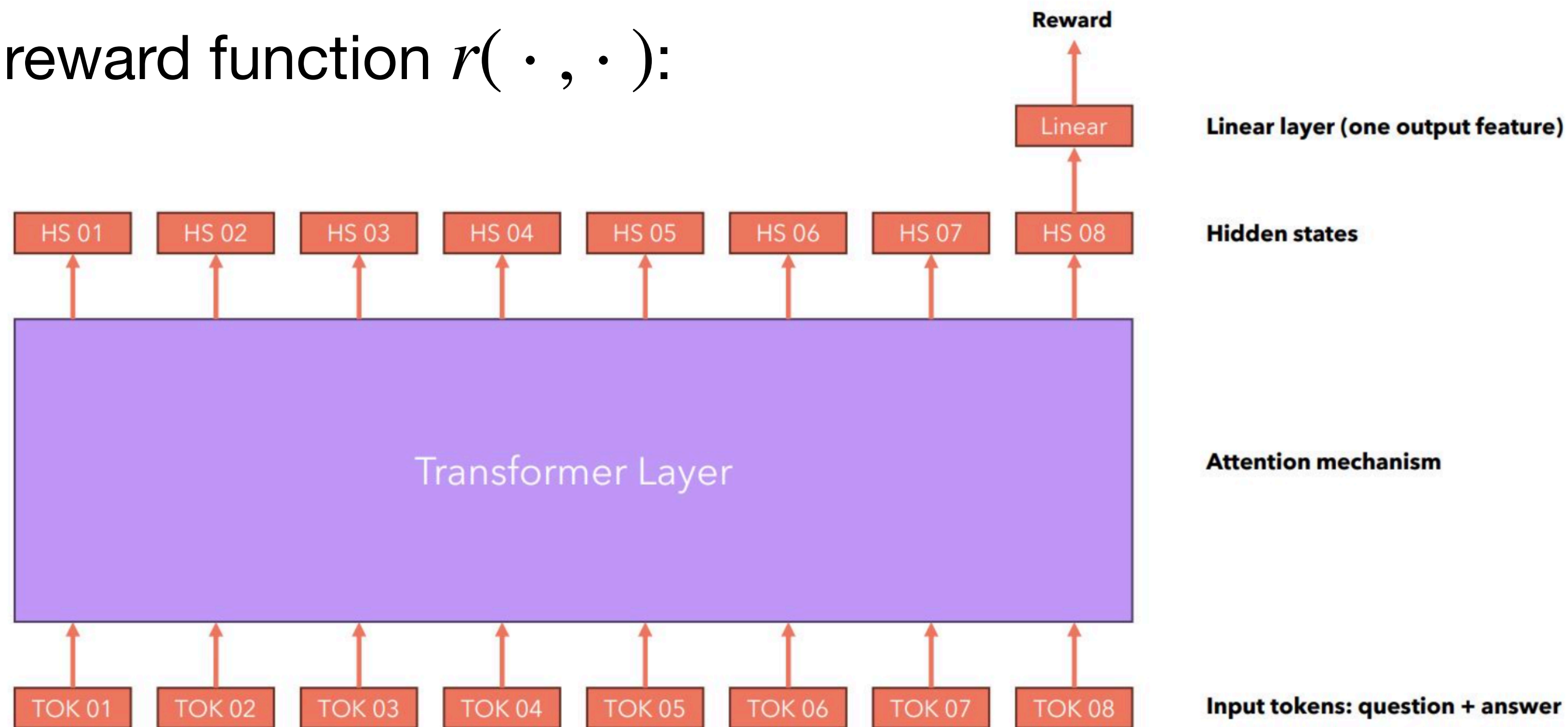
$$= \frac{1}{1 + \exp\{ - (r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2)) \}}$$

- this sigmoid function $\sigma(r^*(x, \hat{x}_1) - r^*(x, \hat{x}_2))$ has many nice properties, including $\log(\sigma(a - b))$ is convex in a and b .

- For us, the probability that we observe a preference ordering $(x, \hat{x}_1, \hat{x}_2, y)$ is

$$\log\left(\mathbb{P}(x, \hat{x}_1, \hat{x}_2, y) \right) = -\log\left(1 + \exp\{ - (\text{sign}(y - 0.5))(r(x, \hat{x}_1) - r(x, \hat{x}_2)) \} \right)$$

- Learning a NN reward function $r(\cdot, \cdot)$:



given data $\mathcal{D} = \{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i)\}$:

Question (Prompt)	Answer 1	Answer 2	Chosen
Where is Shanghai?	Shanghai is a city in China	Shanghai does not exist	1
Explain gravity like I'm 5	Gravity is a famous restaurant	Gravity is what pulls things toward each other. It's why you stay on the ground and planets orbit the sun.	2
What is 2+2?	4	2+2 is a very complicated math problem...	1

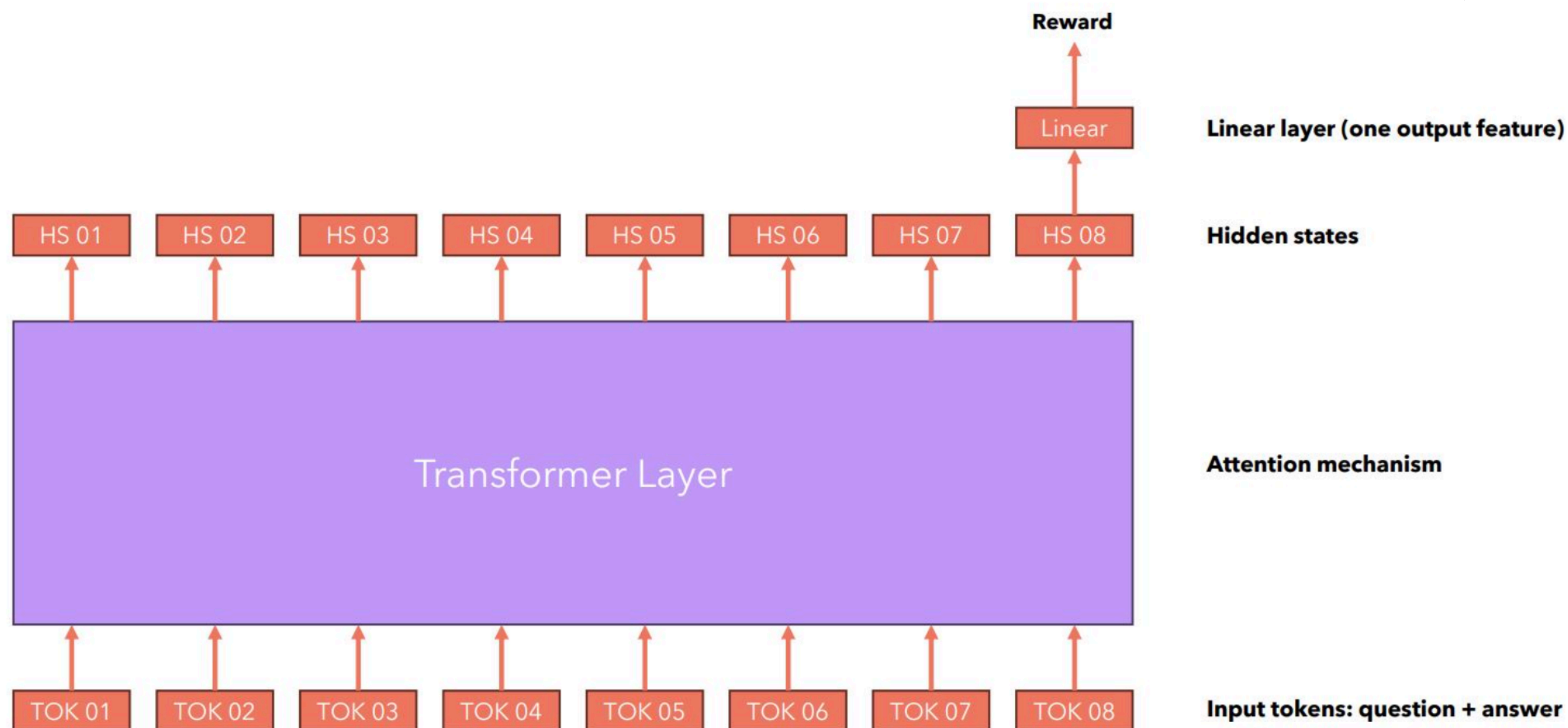
- Training the reward model using **maximum likelihood**:

$$\max_{\text{parameter}} \sum_{i=1}^n \log(\mathbb{P}(\text{sample } i; \text{parameter})) =$$

$$\max_r \sum_{(x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i) \in \mathcal{D}}$$

$$\log \frac{1}{1 + \exp\{ - (\text{sign}(y_i - 0.5))(r(x_i, \hat{x}_{i,1}) - r(x_i, \hat{x}_{i,2})) \}}$$

$$\mathbb{P}((x_i, \hat{x}_{i,1}, \hat{x}_{i,2}, y_i) | \text{reward model } r)$$



- **Chatbot Arena** collects human feedback on completions from pairs of language models, and release them as open RLHF dataset.

Battle ▾ ⚔ Leave Feedback

how does RLHF work?

Assistant A

RLHF stands for Reinforcement Learning from Human Feedback, and it's a technique used to fine-tune AI models (like large language models) to better align with human preferences and values. It's particularly popular in developing safer and more helpful AI systems, such as chatbots. I'll break down how it works step by step.

1. Initial Training (Supervised Fine-Tuning)

- Before RLHF, the AI model is typically pre-trained on a large dataset of text to learn general language patterns.
- Then, it's fine-tuned using supervised learning on a dataset of high-quality examples. These examples are often created by humans, where they provide prompts and the desired responses. This step helps the model generate decent outputs, but it might not yet capture nuanced human preferences (e.g., being helpful, truthful, or non-toxic).

2. Collecting Human Feedback

Assistant B

Reinforcement Learning from Human Feedback (RLHF) is a technique used to align AI models, particularly large language models (LLMs), with human preferences and values. It combines reinforcement learning (RL) with human feedback to fine-tune models for better performance on tasks like generating helpful, honest, and harmless responses. Here's a step-by-step breakdown of how RLHF works:

1. Pretrained Language Model (Base Model)





































- Start with a pretrained language model (e.g., a transformer-based model like GPT) that has been trained on a large corpus of text data using self-supervised learning (e.g., next-token prediction).

2. Supervised Fine-Tuning (SFT)

- The base model is fine-tuned on a smaller, high-quality dataset of human-generated responses to specific prompts. This step helps the model learn to generate more coherent and contextually appropriate

← Left is Better 🤝 It's a tie 🚫 Both are bad Right is Better →

- Also, these pairwise comparisons are used to compute ELO scores for each model, which provides a leaderboard. The breakthrough in Chatbot Arena is that they figure out how to align the incentives for the model providers / the human users / RLHF developers.

Model	Arena Elo	Coding	Vision	AAll	MMLU-Pro	ARC-AGI	Organization	License
 Claude Opus 4.7 Thinking	1505	1565	1310	76	90	75.8	 Anthropic	Proprietary
 Gemini-3.1-Pro	1505	1531	1309	76	91	77.1	 Google	Proprietary
 Claude Opus 4.7	1503	1554	1300	73	89.9	65.5	 Anthropic	Proprietary
 Claude Opus 4.6 Thinking	1503	1545	1304	73	89.7	69.2	 Anthropic	Proprietary
 Grok-4.20	1496	1518	1279	72	89.6	65.1	 xAI	Proprietary
 GPT-5.4-high	1495	1538	1290	73	88.5	74	 OpenAI	Proprietary
 Gemini-3-Pro	1492	1501	1308	73	90	33.6	 Google	Proprietary
 Claude Opus 4.6	1490	1535	1298	71	89.5	64.6	 Anthropic	Proprietary
 Muse Spark	1489	1493	1294	71	87.3		 Meta	Proprietary
 Grok-4.1-Thinking	1482	1483		70	89	26	 xAI	Proprietary
 Gemini-3-Flash	1470	1469	1292	71	89	31.1	 Google	Proprietary
 Claude Opus 4.5 (thinking-32k)	1468	1510		70	89.5	30.6	 Anthropic	Proprietary
 Claude Sonnet 4.6 Thinking	1467	1511	1278	71	88	60.4	 Anthropic	Proprietary
 GLM-5.1 	1467	1506		71	87.1	5.1	 Z.ai	MIT
 Seed2.0 Pro	1466	1495	1288	70	87.8		 ByteDance	Proprietary
 Kimi-K2.6-Thinking 	1466	1493	1286	71	87.3		 Moonshot	Modified MIT
 Qwen3.5-Max	1466	1493		70	87.8		 Alibaba	Proprietary

- **RLHF:** Strategy for further training the LM to make it aligned.
 1. Pretrain a language model.
 2. Learn a reward model $r(x, \hat{x})$ that maps arbitrary prompt and completion to a real value
 3. Fine-tune a pre-trained LM with standard Reinforcement Learning (RL) using the reward model.
- This is one example of RLHF and there are several difference variations, including Direct Preference Optimization (DPO) and Group Relative Policy Optimization (GRPO).

- **Policy update:** We want to train the LM parameters w to maximize the reward:

$$J(w) = \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x})]$$

where **prompt** x is drawn from some natural distribution of the prompts, and the **completion** \hat{x} is drawn from **the policy** π_w **of the language model**. This is the expected reward when sampling from the LM.

- The optimization problem we want to solve is

$$\max_w J(w) = \max_w \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x})]$$

- We use iterative algorithms such as **gradient ascent** to solve this:

$$w \leftarrow w + \alpha \nabla_w J(w)$$

but the gradient is taken with respect to to the sampling distribution.

- A very brief overview of **policy gradient** to compute the gradient:
- We need to compute

$$\begin{aligned}
 \nabla_w J(w) &= \nabla_w \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x})] \\
 &= \nabla_w \left\{ \sum_{x, \hat{x}} r(x, \hat{x}) p(x) \pi_w(\hat{x} | x) \right\} \\
 \text{(linearity of expectation)} \quad &= \sum_{x, \hat{x}} \left\{ r(x, \hat{x}) p(x) \nabla_w \pi_w(\hat{x} | x) \right\}
 \end{aligned}$$

- how do we get this to form that looks like

$$= \sum_{x, \hat{x}} \left\{ p(x) \pi_w(\hat{x} | x) \cdots \right\}$$

- A very brief overview of **policy gradient** to compute the gradient:
- We need to compute

$$\begin{aligned} \nabla_w J(w) &= \nabla_w \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x})] \\ &= \nabla_w \left\{ \sum_{x, \hat{x}} r(x, \hat{x}) p(x) \pi_w(\hat{x} | x) \right\} \end{aligned}$$

(linearity of expectation)

$$= \sum_{x, \hat{x}} \left\{ r(x, \hat{x}) p(x) \nabla_w \pi_w(\hat{x} | x) \right\}$$

- **log-derivative trick:** $\nabla_w \{ \log(\pi_w(\hat{x} | x)) \} = \frac{1}{\pi_w(\hat{x} | x)} \nabla_w \pi_w(\hat{x} | x)$

- A very brief overview of **policy gradient** to compute the gradient:
- We need to compute

$$\begin{aligned} \nabla_w J(w) &= \nabla_w \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x})] \\ &= \nabla_w \left\{ \sum_{x, \hat{x}} r(x, \hat{x}) p(x) \pi_w(\hat{x} | x) \right\} \end{aligned}$$

(linearity of expectation)

$$= \sum_{x, \hat{x}} \left\{ r(x, \hat{x}) p(x) \nabla_w \pi_w(\hat{x} | x) \right\}$$

- **log-derivative trick:** $\nabla_w \{ \log(\pi_w(\hat{x} | x)) \} = \frac{1}{\pi_w(\hat{x} | x)} \nabla_w \pi_w(\hat{x} | x)$

$$\begin{aligned} &= \sum_{x, \hat{x}} \left\{ r(x, \hat{x}) p(x) \pi_w(\hat{x} | x) \nabla_w \log(\pi_w(\hat{x} | x)) \right\} \\ &= \mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} [r(x, \hat{x}) \nabla \log(\pi_w(\hat{x} | x))] \end{aligned}$$

- Now that the gradient is inside the expectation, we can use samples to approximate it

$$\mathbb{E}_{(x, \hat{x}) \sim D_{\pi_w}} \left[r(x, \hat{x}) \nabla \log(\pi_w(\hat{x} | x)) \right] = \frac{1}{m} \sum_{i=1}^m r(x_i, \hat{x}_i) \nabla_w \log(\pi_w(\hat{x}_i | x_i))$$

- The update rule is

$$w_{t+1} \leftarrow w_t + \alpha \frac{1}{m} \sum_{i=1}^m r(x_i, \hat{x}_i) \nabla \log(\pi_w(\hat{x} | x))$$

- But training a reward model can be challenging.

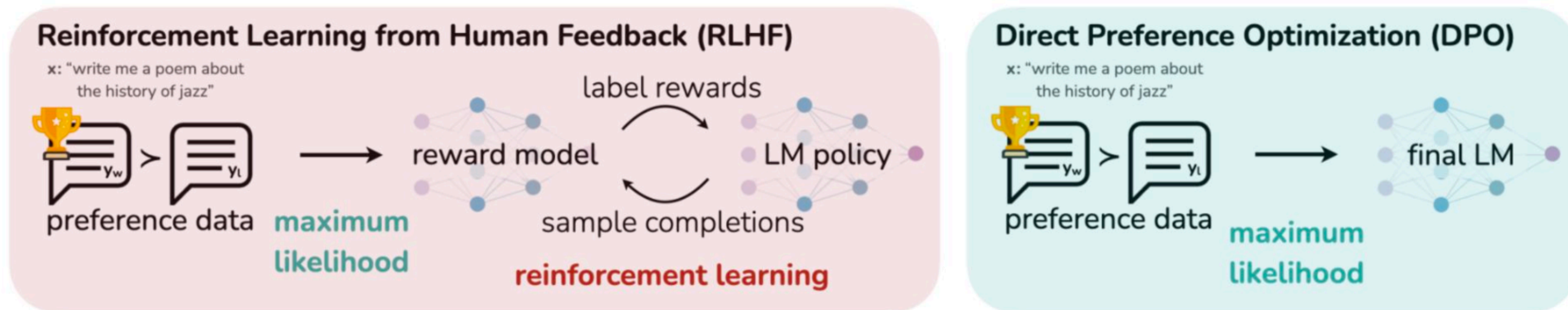


Figure 1: **DPO optimizes for human preferences while avoiding reinforcement learning.** Existing methods for fine-tuning language models with human feedback first fit a reward model to a dataset of prompts and human preferences over pairs of responses, and then use RL to find a policy that maximizes the learned reward. In contrast, DPO directly optimizes for the policy best satisfying the preferences with a simple classification objective, fitting an *implicit* reward model whose corresponding optimal policy can be extracted in closed form.

- **DPO** (Direct Preference Optimization) defines the Bradley-Terry model directly from the policy:

$$\mathbb{P}(\hat{x}_1 \succ \hat{x}_2)(w) = \frac{1}{1 + \exp \left\{ \log \left(\frac{\pi_w(\hat{x}_2 | x)}{\pi_{\text{ref}}(\hat{x}_2 | x)} \right) - \log \left(\frac{\pi_w(\hat{x}_1 | x)}{\pi_{\text{ref}}(\hat{x}_1 | x)} \right) \right\}}$$

- **RLVR (Reinforcement Learning with Verifiable Reward)** assumes that there is an oracle that can give an exact reward. For example in **math** or **coding**, one can have a verifier that can assert correctness.
- RLVR and the verifier can be used to train a **reasoning model** that can iteratively solve complex reasoning tasks at inference time. This is called **inference time compute** or **inference time scaling**:
how much better can you do if you can spend more compute/time at inference?
- example 1. card arithmetic reasoning task

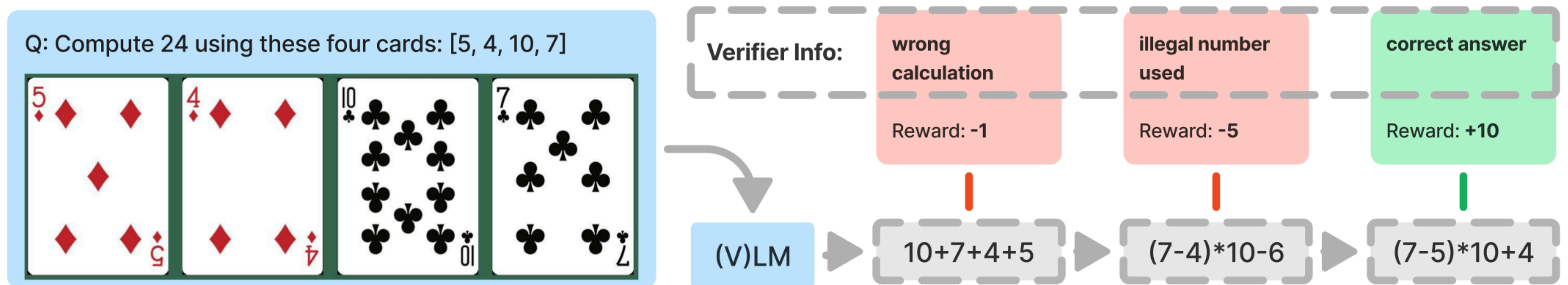


Figure 2: An example of the sequential revision formulation with a verifier. The model generate the next answer v_{t+1}^{out} conditioned on all previous answers and information $(v_i^{\text{out}}, v_t^{\text{ver}}, 0 \leq i \leq t)$ from the verifier.

- example 2.
visual navigation
reasoning task



★ First, **turn slightly right** towards the northeast and walk a short distance until you reach the next intersection, where you'll see **The Dutch** on your right. Next, make a **sharp left turn** to head northwest. Continue for a while until you reach the next intersection, where **Lola Taverna** will be on your right. Finally, **turn slightly right** to face northeast and walk a short distance until you reach your destination, **Shuka**, which will be on your right.

Figure 4: **Demonstration of one navigation task in V-IRL.** Agent navigates from place to place following the given linguistic navigation instructions in V-IRL. The navigation procedure is shown at the top, with the navigation instructions displayed below. Visual observation-related information is highlighted in **green**, while action-related information is marked in **orange**.

- “**SFT Memorizes, RL Generalizes: A Comparative Study of Foundation Model Post-training**” [Chu et al. 2025]
- **SFT**: train on optimal solution.
- **RL**: train policy on a reward model.
- evaluated on **OOD variation**:
 - card: J,Q,K are {10,10,10} or {11,12,13}
 - navigation: N/E/S/W vs. right/left

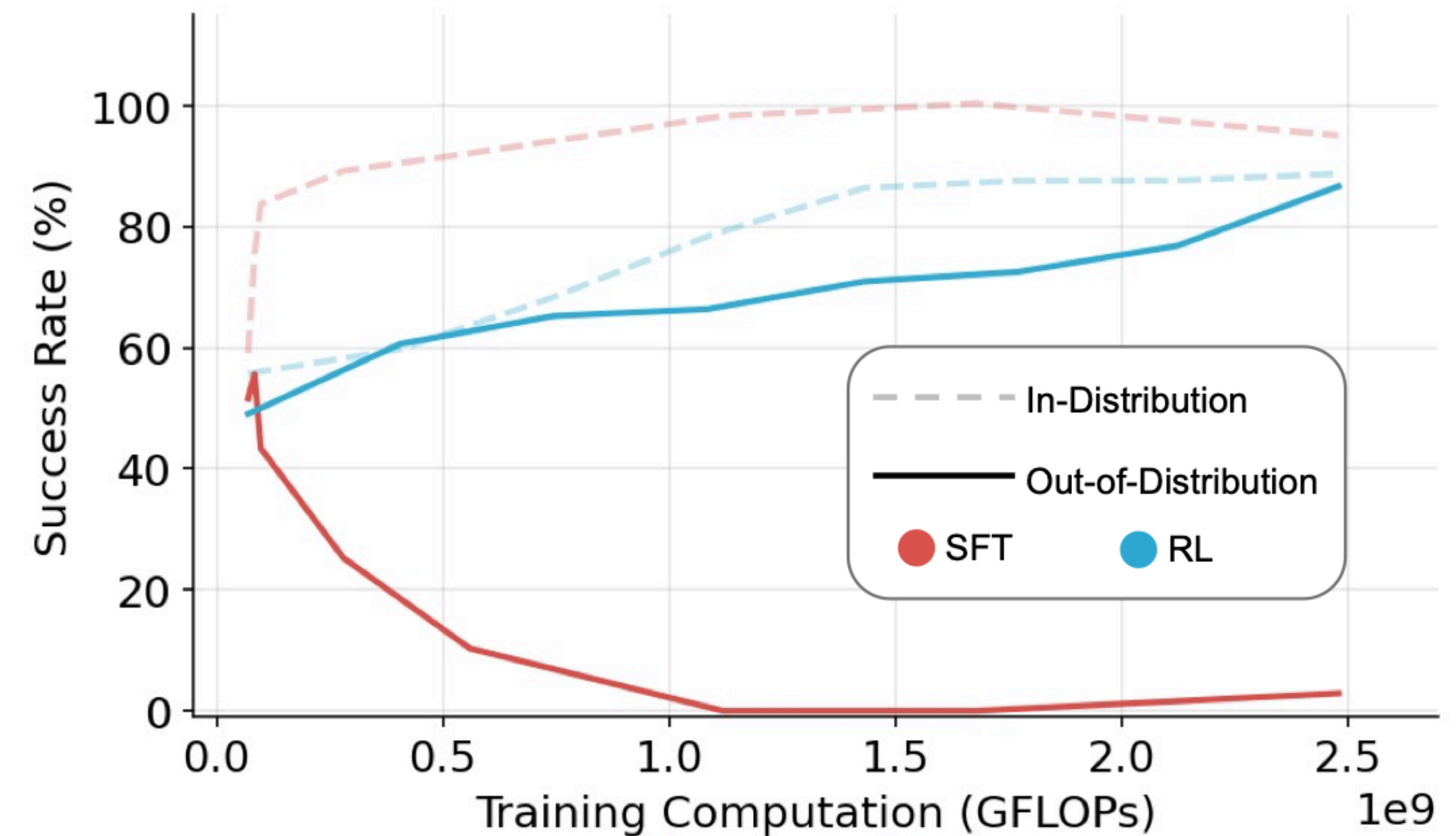


Figure 1: A comparative study of RL and SFT on the visual navigation environment $V\text{-IRL}$ (Yang et al., 2024a) for OOD generalization. OOD curves represent performance on the same task, using a *different textual action space*. See detailed descriptions of the task in Section 5.1.

- “Reinforcement Learning for Reasoning in Large Language Models with One Training Example” [Wang et al. 2025] <https://arxiv.org/pdf/2504.20571> demonstrates **extreme generalization**.

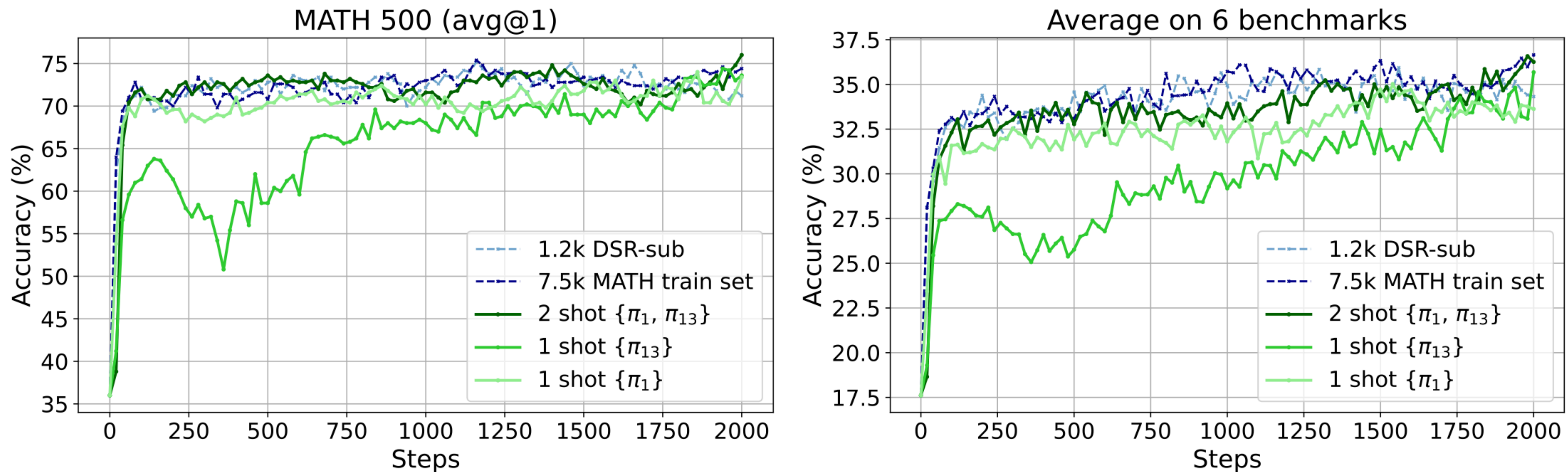


Figure 1: **RLVR with 1 example (green) can perform as well as using datasets with thousands of examples (blue)**. Left/Right corresponds to MATH500/Average performance on 6 mathematical reasoning benchmarks (MATH500, AIME24, AMC23, Minerva Math, OlympiadBench, and AIME25). Base model is **Qwen2.5-Math-1.5B**. π_1 and π_{13} are examples defined by Eqn. 2 and detailed in Tab. 2, and they are from the 1.2k DeepScalerR subset (DSR-sub). Setup details are in Sec. 3.1. We find that RLVR with 1 example $\{\pi_{13}\}$ (35.7%) performs close to that with 1.2k DSR-sub (35.9%), and RLVR with 2 examples $\{\pi_1, \pi_{13}\}$ (36.6%) even performs better than RLVR with DSR-sub and as well as using 7.5k MATH train dataset (36.7%). Detailed results are in Fig. 6 in Appendix C.1.1. Additional

- So which example is special?

Table 2: **Example** π_1 . It is from DSR-sub (Sec. 3.1). A more precise answer should be "12.7".

Prompt of example π_1 :

The pressure (P) exerted by wind on a sail varies jointly as the area (A) of the sail and the cube of the wind's velocity (V) . When the velocity is (8) miles per hour, the pressure on a sail of (2) square feet is (4) pounds. Find the wind velocity when the pressure on (4) square feet of sail is (32) pounds. Let's think step by step and output the final answer within `\boxed{}`.

Ground truth (label in DSR-sub): 12.8.

- key step is obtaining $k = 1/256$ for formula $P = kAV^3$, and calculating $V = (2048)^{1/3}$

- **Group Relative Policy Optimization (GRPO)** is used, which is introduced by DeepSeek-R1

- multiple completions $\{\hat{x}_i\}_{i=1}^G$ are collected from the model policy on a single prompt x

$$J(w) = \frac{1}{G} \sum_{i=1}^G \left\{ \min \left\{ A_i \frac{\pi_w(\hat{x}_i | x)}{\pi_{\text{ref}}(\hat{x}_i | x)}, A_i \text{Clip} \left(\frac{\pi_w(\hat{x}_i | x)}{\pi_{\text{ref}}(\hat{x}_i | x)}, 1 - \varepsilon, 1 + \varepsilon \right) \right\} \right\}$$

where the **group relative Advantage for sample i** is defined as

$$A_i := \frac{r_i - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}$$

and **Clipping** provides some regularization s.t. no one sample dominates

- “Spurious Rewards: Rethinking Training Signals in RLVR” [Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang et al.] <https://arxiv.org/abs/2506.10947>.

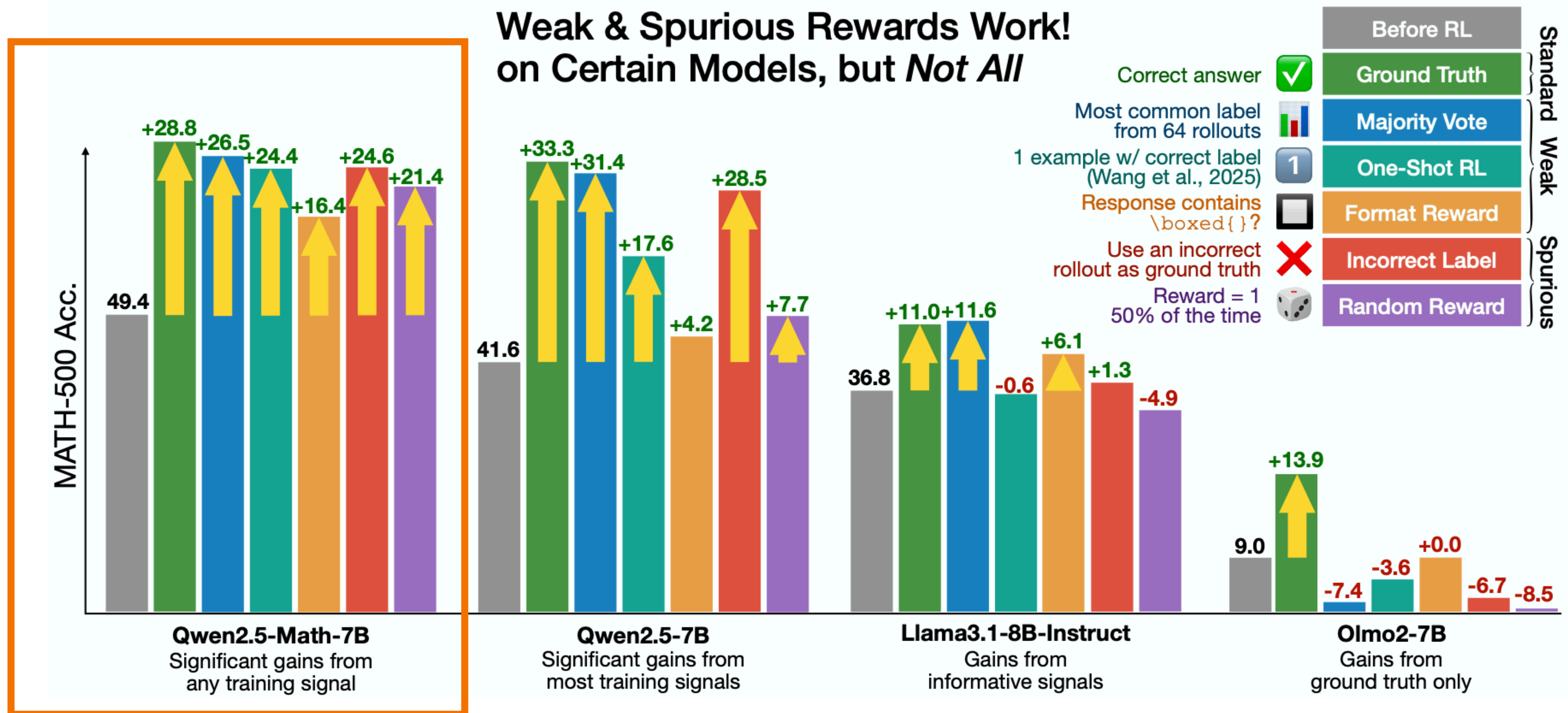


Figure 1: MATH-500 accuracy after 150 steps of RLVR on various training signals. We show that even “spurious rewards” (e.g., rewarding *incorrect* labels or with completely random rewards) can yield strong MATH-500 gains on Qwen models. Notably, these reward signals do not work for other models like Llama3.1-8B-Instruct and OLMo2-7B, which have different reasoning priors.

- Where is the gain coming from?
Code reasoning: Coding helps in reasoning despite no access to code interpreter

MATH Question:

What is the distance, in units, between the points $(2, -6)$ and $(-4, 3)$? Express your answer in simplest radical form.

Qwen2.5-Math-7B Solution (correct):

To find the distance between two points (x_1, y_1) and (x_2, y_2) in a Cartesian plane... Let's break this down step-by-step and compute the result using Python.

```

1 import math
2 ...
3 # Calculate the distance using the distance formula
4 distance = math.sqrt(dx**2 + dy**2)
5 print(distance)

```

output: 10.816653826391969
 ...
 Thus, the final answer is: $3\sqrt{13}$

Figure 5: Example of Qwen2.5-Math-7B's code reasoning (see Figure 19 for the complete response). The question is randomly picked from the MATH-500 test set. Note that both the code and the code execution result are autoregressively generated by Qwen2.5-Math-7B. **No external code interpreter was provided to the model.**

Model	Qwen2.5-Math-7B	Qwen2.5-Math-1.5B	Qwen2.5-7B	OLMo2-7B-SFT
Code Frequency	65.0	53.6	92.2	98.0
Acc. w/ Code	60.9	52.6	39.9	21.0
Acc. w/ Lang	35.0	17.2	61.5	40.0

RL with random reward encourages more code reasoning (65% → 95%), and better code reasoning models benefit more

- RL with random reward encourages Qwen models to use more code, leveraging Qwen’s code reasoning capability. Other models that do not have code reasoning capability do not gain as much from random rewards.

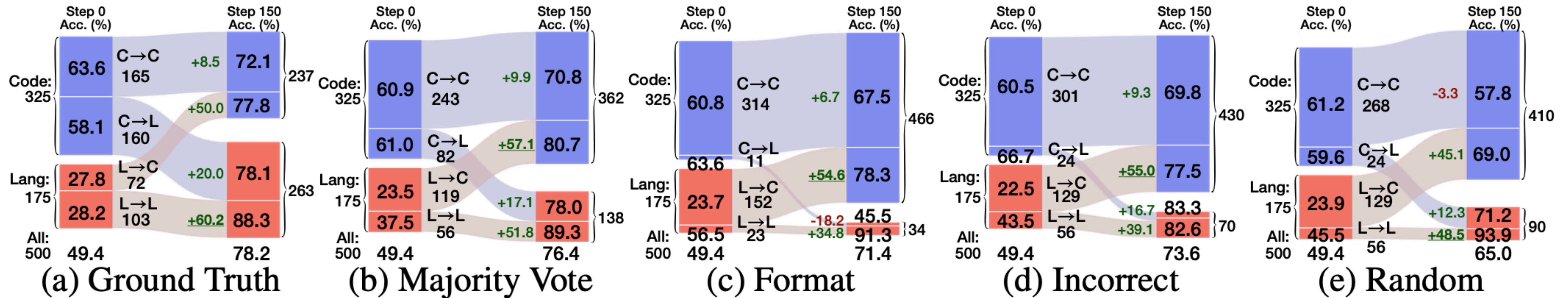


Table 2: Partial contribution to the overall performance gain averaged over rewards that successfully steered the model’s reasoning strategy (Figure 6).

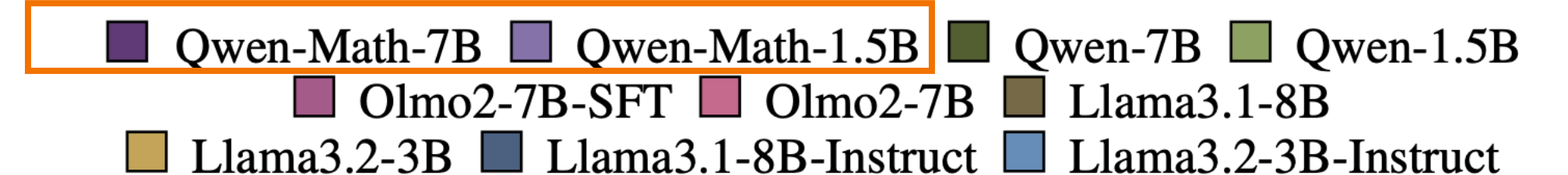
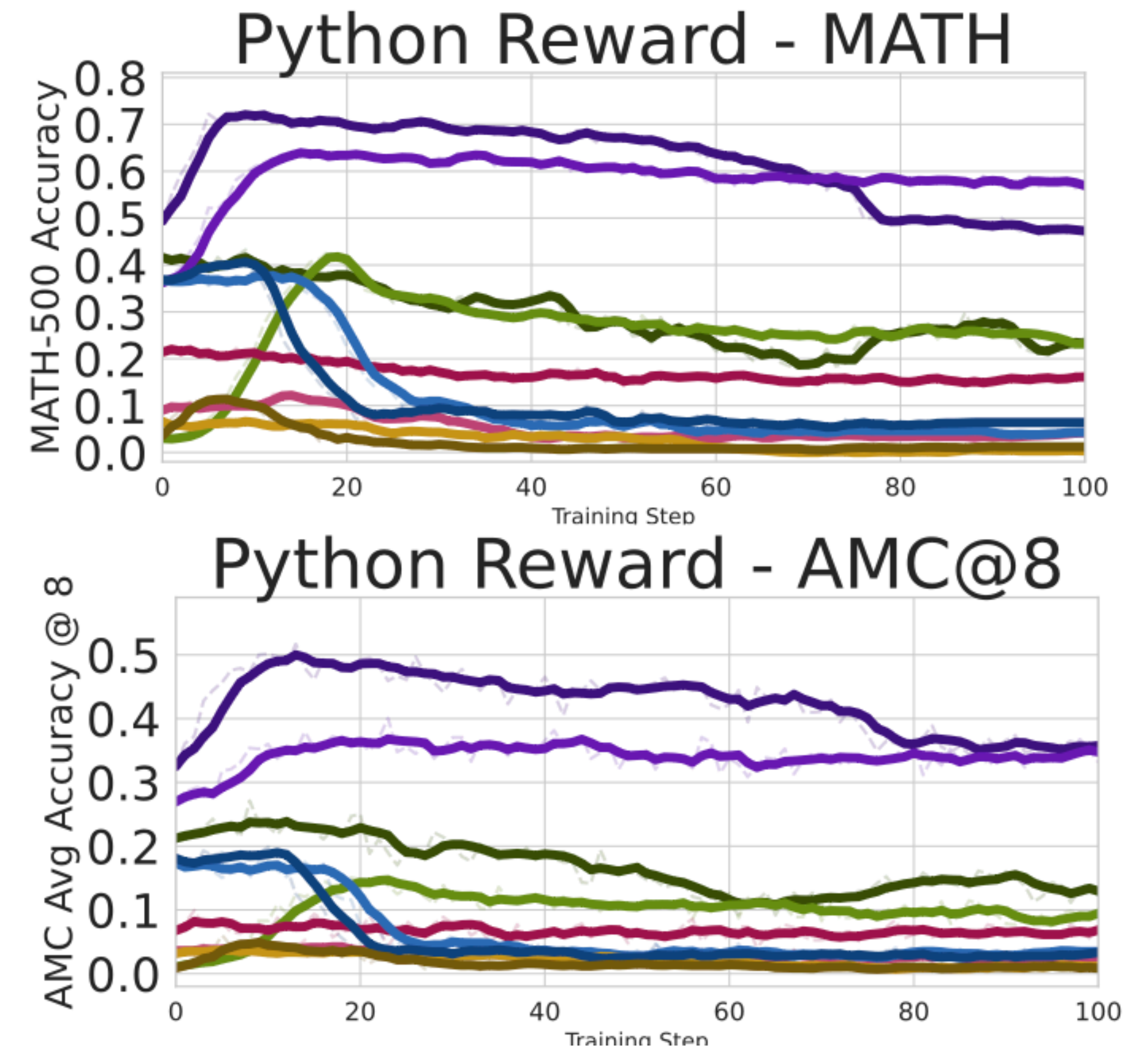
Model	Qwen2.5-Math-7B	Qwen2.5-Math-1.5B	Qwen2.5-7B
Avg. Total Gain	↑ 23.5%	↑ 28.5%	↑ 30.6%
$C_{Code \rightarrow Code}$	11.6%	2.8%	5.0%
$C_{Code \rightarrow Lang}$	8.6%	2.0%	93.9%
$C_{Lang \rightarrow Code}$	58.3%	78.7%	0.0%
$C_{Lang \rightarrow Lang}$	21.4%	16.5%	5.9%

Qwen2.5-7B does not know how to reason in code

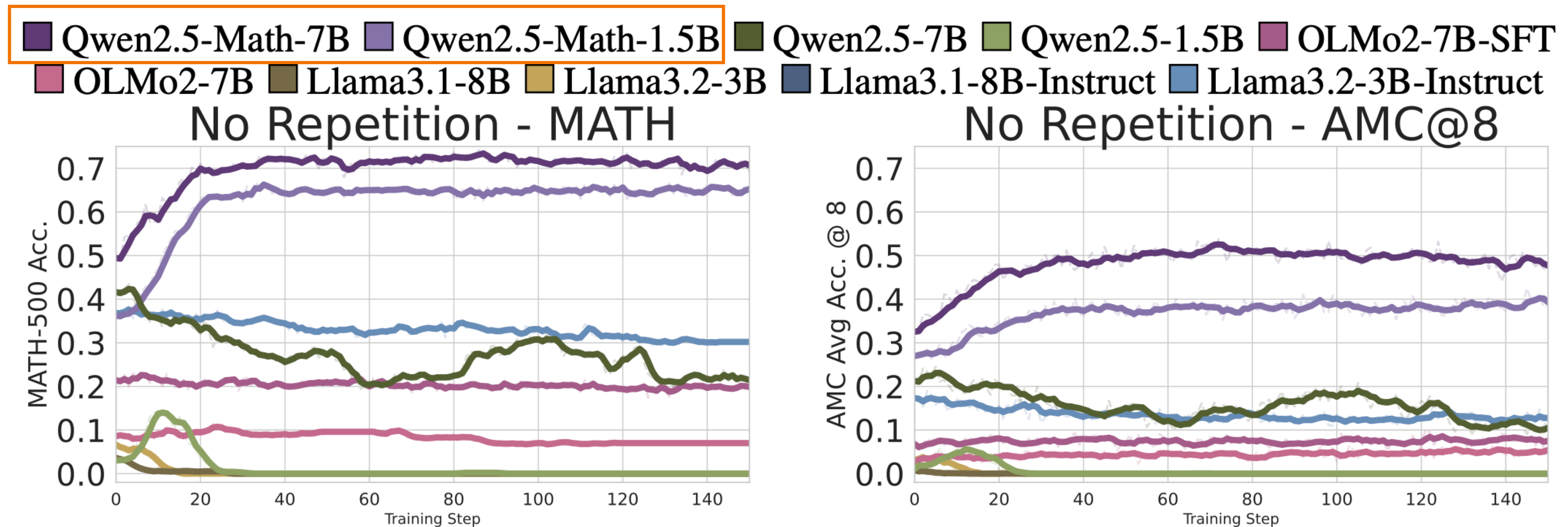
- prompting with “Let’s solve this using Python.” to force the model to use code and measuring accuracy change:

Model	Original	Prompting	Abs. Diff.
Qwen2.5-Math-1.5B	34.8%	60.4%	+25.6%
Qwen2.5-Math-7B	52.6%	64.4%	+11.8%
Qwen2.5-1.5B	2.8%	13.0%	+10.2%
Qwen2.5-7B	42.8%	22.2%	-20.6%
Llama3.2-3B-Instruct	36.6%	8.2%	-28.4%
Llama3.1-8B-Instruct	38.4%	15.2%	-23.2%
OLMo2-7B	10.2%	7.8%	-2.4%
OLMo2-7B-SFT	20.4%	18.6%	-1.8%

- RL with Python reward:



- Another feature correlated with better performance: no repetition.
- If we run RL that rewards having less repetition, Qwen-Math models improve significantly.

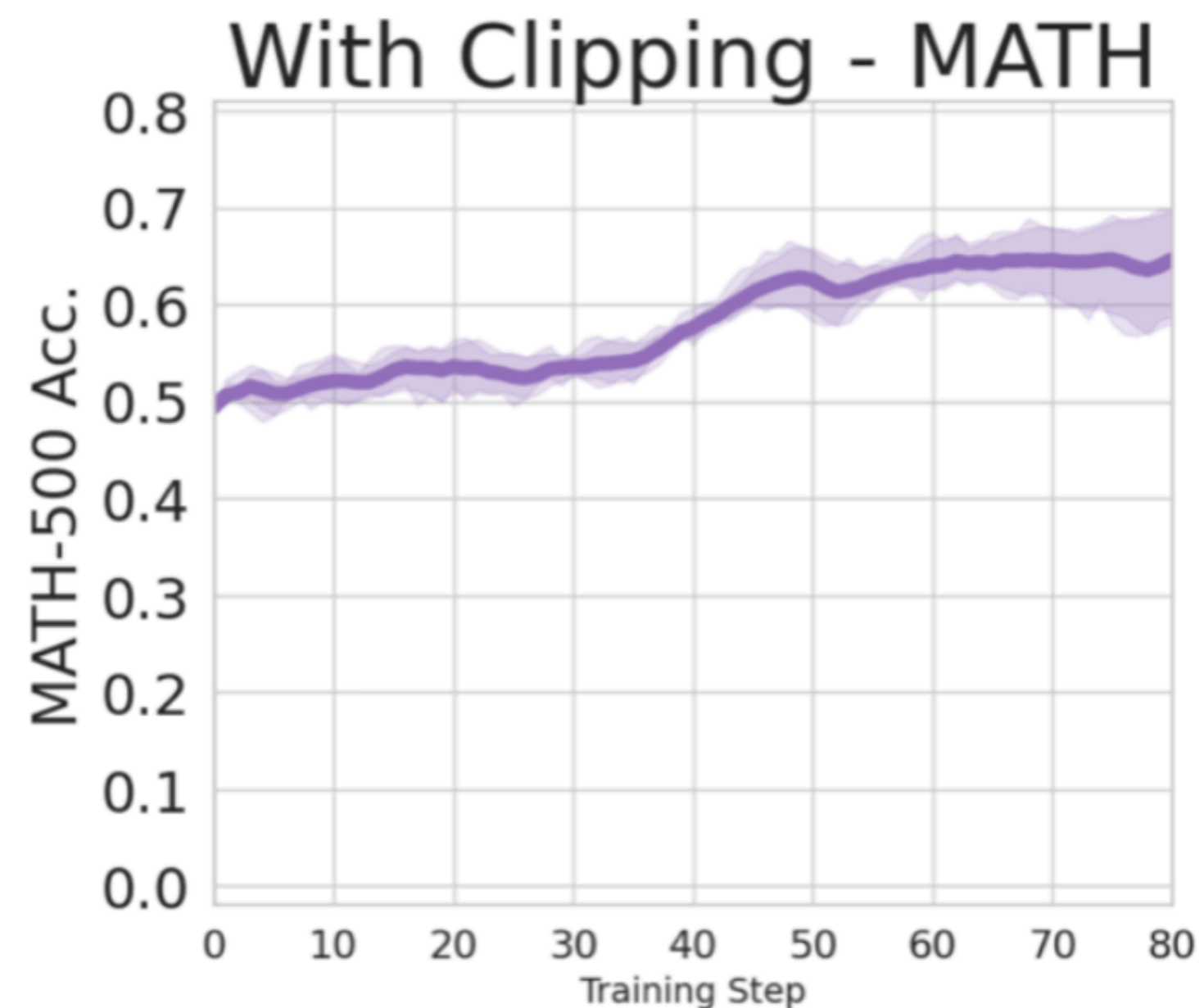
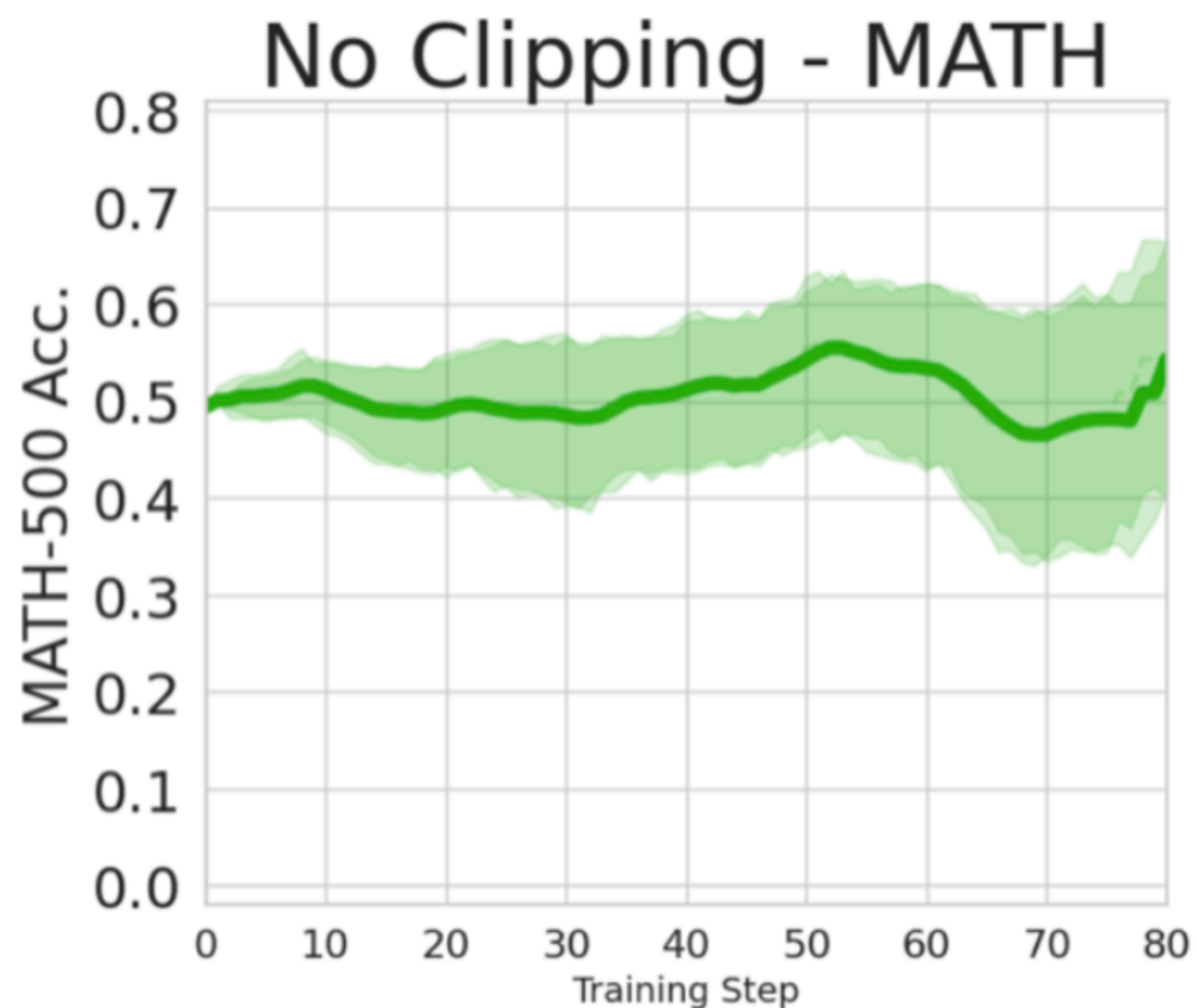


- We suspect that there are several spurious features that correlate with better reasoning, and RL with random rewards can encourage some of those features.

- Where does the gain come from?
- Recall **Group Relative Policy Optimization (GRPO)** is used,

$$J(w) = \frac{1}{G} \sum_{i=1}^G \left\{ \min \left\{ A_i \frac{\pi_w(\hat{x}_i | x)}{\pi_{\text{ref}}(\hat{x}_i | x)}, A_i \text{Clip} \left(\frac{\pi_w(\hat{x}_i | x)}{\pi_{\text{ref}}(\hat{x}_i | x)}, 1 - \varepsilon, 1 + \varepsilon \right) \right\} \right\}$$

where $A_i = \frac{r_i - \text{mean}(r_1, r_2, \dots, r_G)}{\text{std}(r_1, r_2, \dots, r_G)}$



$$\text{Bias}(\nabla_{\theta}L(\theta)) = \mathbb{E}[\nabla_{\theta}L(\theta)] - \mathbb{E}[\nabla_{\theta}L^{\text{unclipped}}(\theta)] = \mathbb{E}[\nabla_{\theta}L(\theta)].$$

$$= \mu \cdot \mathbb{E}_{x,y} \left[\begin{cases} \nabla_{\theta}R_{\theta}, & \text{if } \pi_{\theta,x}(y_t) < \pi_{\text{old},x}(y_t) \cdot (1 - \epsilon_c), \\ 0, & \text{if } \pi_{\text{old},x}(y_t) \cdot (1 - \epsilon_c) \leq \pi_{\theta,x}(y_t) \\ & \leq \pi_{\text{old},x}(y_t) \cdot (1 + \epsilon_c), \\ -\nabla_{\theta}R_{\theta}, & \text{if } \pi_{\theta,x}(y_t) > \pi_{\text{old},x}(y_t) \cdot (1 + \epsilon_c). \end{cases} \right]$$

Sources

- Other courses in LLMs that the lecture slides are based on
 - CSE493S/599S at UW by Ludwig Schmidt: <https://mlfoundations.github.io/advancedml-sp23/>
 - EE-628 at EPFL by Volkan Cevher: <https://www.epfl.ch/labs/lions/teaching/ee-628-training-large-language-models/ee-628-slides-2025/>
 - <https://sharif-llm.ir/assets/lectures/Chain-of-Thought-Prompting.pdf>
 - <https://www.cs.princeton.edu/courses/archive/fall22/cos597G/lectures/lec09.pdf>
- Useful blog posts
 - <https://azizbelaweid.substack.com/p/complete-summary-of-absolute-relative>
 - <https://blog.dust.tt/speculative-sampling-llms-writing-a-lot-faster-using-other-llms/>
 - <https://gordicaleksa.medium.com/eli5-flash-attention-5c44017022ad>
 - <https://medium.com/@dilliprasad60/qlora-explained-a-deep-dive-into-parametric-efficient-fine-tuning-in-large-language-models-llms-c1a4794b1766>
- Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft). draft, third edition, 2023.
- Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space”, In International Conference on Learning Representations, 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation”, Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
- Ofir Press, Noah A. Smith^{1,3} Mike Lewis², “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation”, In International Conference on Learning Representations, 2022
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”, In Neural Information Processing Systems, 2017
- Beitong Zhou, Cheng Cheng, Guijun Ma, and Yong Zhang. “Remaining useful life prediction of lithium-ion battery based on attention mechanism with positional encoding”, In IOP Conference Series: Materials Science and Engineering, 2020.
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks.” In International Conference on Machine Learning, 2013

- Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In *Neural Computation*, 9(8):1735–1780, 11 1997.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning phrase representations using rnn encoder-decoder for statistical machine translation”, In *ACL 2014*
- Andrey Andreyevich Markov. “Essai d’une recherche statistique sur le texte du roman. ‘Eugene Onegin’ illustrant la liaison des epreuve en chain”. In: *Izvestia Imperatorskoi Akademii Nauk (Bulletin de l’Académie Impériale des Sciences de St.-Pétersbourg)*. 6th ser, 7:153–162, 1913.
- Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, Yejin Choi, “The Curious Case of Neural Text Degeneration”, In *International Conference on Learning Representations, 2020*
- Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre and John Jumper, “Accelerating Large Language Model Decoding with Speculative Sampling” In, *ACL-findings, 2024*
- Sergey Ioffe, Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”, In *International Conference on Machine Learning, 2015*
- Shibani Santurkar* MIT shibani@mit.edu Dimitris Tsipras* MIT tsipras@mit.edu Andrew Ilyas* MIT ailyas@mit.edu Aleksander Madry, “How Does Batch Normalization Help Optimization?”, In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*
- Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, “Layer Normalization “, In 2016
- Tianyu Gao, Adam Fisch, Danqi Chen, “Making Pre-trained Language Models Better Few-shot Learners”, In *ACL, 2021*
- Sewon Min^{1,2} Xinxi Lyu¹ Ari Holtzman¹ Mikel Artetxe² Mike Lewis² Hannaneh Hajishirzi^{1,3} Luke Zettlemoyer, “rethinking the role of demonstrations what makes in conte...”
- Hila Gonen^{1,2} Srini Iyer² Terra Blevins¹ Noah A. Smith^{1,3} Luke Zettlemoyer¹, “Demystifying Prompts in Language Models via Perplexity Estimation”
- E Akyürek, B Wang, Y Kim, J Andreas , “In-context language learning: Architectures and algorithms”, 2024
- What learning algorithm is in-context learning? Investigations with linear models Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, Denny Zhou, 2022
- Ziqian Lin, Kangwook Lee, “Dual Operating Modes of In-Context Learning”, 2024
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models”, In *NeurIPS 2022*
- Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa, “Large Language Models are Zero-Shot Reasoners”, In *NeurIPS 2022*
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, Denny Zhou, “Self-Consistency Improves Chain of Thought Reasoning in Language Models”, In *ICLR 2023*

- Shunyu Yao · Jeffrey Zhao · Dian Yu · Nan Du · Izhak Shafran · Karthik Narasimhan, Yuan Cao, “ReAct: Synergizing Reasoning and Acting in Language Models”, In ICLR 2025
- Satyapriya Krishna¹, Kalpesh Krishna², Anhad Mohananey^{†2}, Steven Schwarcz², Adam Stambler², Shyam Upadhyay², Manaal Faruqui^{*3}, “Fact, Fetch, and Reason: A Unified Evaluation of Retrieval-Augmented Generation“
- Salaheddin Alzubi, Creston Brooks, Purva Chiniya, Edoardo Contente, Chiara von Gerlach, Lucas Irwin, Yihan Jiang, Arda Kaz, Windsor Nguyen, Sewoong Oh, Himanshu Tyagi, Pramod Viswanath, “Open Deep Search: Democratizing Search with Open-source Reasoning Agents“, <https://arxiv.org/abs/2503.20201>
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, Heng Ji, “Executable Code Actions Elicit Better LLM Agents”, In *ICML 2024*
- Peter Shaw, Jakob Uszkoreit Ashish Vaswani, “Self-Attention with Relative Position Representations”, 2018
- Ofir Press^{1,2} Noah A. Smith^{1,3} Mike Lewis², “TRAIN SHORT, TEST LONG: ATTENTION WITH LINEAR BIASES ENABLES INPUT LENGTH EXTRAPOLATION”, 2022
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, “QLoRA: Efficient Finetuning of Quantized LLMs”, In NeurIPS 2023
- Fanxu Meng^{1,2}, Zhaohui Wang¹, Muhan Zhang^{1,2*}, “PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models” In *NeurIPS 2024*
- Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, Yun Chen, “MiLoRA: Harnessing Minor Singular Components for Parameter-Efficient LLM Finetuning”, In *NAACL 2025*
- Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, Jacek Tabor, “LoRA-XS: Low-Rank Adaptation with Extremely Small Number of Parameters”
- Bingcong Li, Liang Zhang, Aryan Mokhtari, Niao He, “On the Crucial Role of Initialization for Matrix Factorization.”, In *ICLR 2025*
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. “Scaling laws for neural language models”, 2020.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

- Sean McLeish, John Kirchenbauer, David Yu Miller, Siddharth Singh, Abhinav Bhatele, Micah Goldblum, Ashwinee Panda, and Tom Goldstein. “Gemstones: A model suite for multi-faceted scaling laws”, 2025

-